Efficient Online Structured Output Learning for Keypoint-Based Object Tracking

Sam Hare[1] Amir Saffari[1,2] Philip H. S. Torr[1]
[1]Oxford Brookes University, Oxford, UK [2]Sony Computer Entertainment Europe, London, UK
{sam.hare,philiptorr}@brookes.ac.uk amir@ymer.org

Abstract

Efficient keypoint-based object detection methods are used in many real-time computer vision applications. These approaches often model an object as a collection of keypoints and associated descriptors, and detection then involves first constructing a set of correspondences between object and image keypoints via descriptor matching, and subsequently using these correspondences as input to a robust geometric estimation algorithm such as RANSAC to find the transformation of the object in the image. In such approaches, the object model is generally constructed offline, and does not adapt to a given environment at runtime. Furthermore, the feature matching and transformation estimation stages are treated entirely separately. In this paper, we introduce a new approach to address these problems by combining the overall pipeline of correspondence generation and transformation estimation into a single structured output learning framework.

Following the recent trend of using efficient binary descriptors for feature matching, we also introduce an approach to approximate the learned object model as a collection of binary basis functions which can be evaluated very efficiently at runtime. Experiments on challenging video sequences show that our algorithm significantly improves over state-of-the-art descriptor matching techniques using a range of descriptors, as well as recent online learning based approaches.

1. Introduction

Keypoint-based object detection has become a cornerstone of modern computer vision, enabling great advances in areas such as augmented reality (AR) and simultaneous localization and mapping (SLAM). These object detection approaches model an object as a set of keypoints, which are matched independently in an input image. Robust estimation procedures based on RANSAC [4, 5, 16] are then used to determine geometrically consistent sets of matches which can be used to infer the presence and transformation of the object.

There has been a great deal of progress in making these approaches suitable for real-time applications, and there are now a range of methods available for use on a desktop PC [1, 9, 12]. Recently, there has been significant interest in developing approaches suitable for low-powered mobile devices such as smartphones and tablets, which are becoming increasingly popular platforms for computer vision applications [3, 10, 14, 15]. These approaches focus on making the matching stage as efficient as possible, since this is generally the most time-consuming part of the detection pipeline. To achieve this, they design image descriptors which can be represented as binary vectors, allowing matching to be performed very efficiently by measuring Hamming distance between descriptors, which can be implemented using binary CPU instructions.

The object models built by traditional approaches are static, usually constructed offline for a particular object. For certain applications like AR and SLAM, however, we want to detect the object repeatedly in a dynamic environment. Additionally, some applications require *on-the-fly* learning and detection to build an instantaneous model from only a single snapshot of the object. Therefore it is desirable to be able to learn an object model efficiently online and adapt it to a particular environment, which is not typically addressed by traditional approaches. This process of adapting or learning the model should not add significant overhead to the detection pipeline, and should still be suitable for real-time detection on low-powered devices. These requirements create a very challenging problem for a learning algorithm.

The approach we propose in this paper frames the entire object detection procedure as structured output prediction, such that overall detection performance can be optimized given a set of training images. Our formulation combines feature learning, matching, and pose estimation into a single unified framework. Furthermore, because we use a linear structured SVM to perform learning, we are able to perform training online, which allows us to quickly adapt our model

to a given environment. Additionally, we show that we can accurately approximate our model during evaluation in such a way that we can take advantage of binary descriptors and the efficiency they provide. As a result, our algorithm adds a relatively small amount of computational overhead compared to static models, while improving the detection rate significantly.

2. Motivation and related work

Keypoint-based methods for geometric object detection generally follow a two stage approach:

1. Finding a set of 2D correspondences between an object model and an input image.

2. Estimating the transformation of the object in the image using a robust geometric verification method based on hypotheses generated from the correspondences (*e.g.* RANSAC and its variants).

Generally these two steps are considered as separate problems, and many algorithms focus on improving the object detection quality by employing robust methods for each of these steps individually.

To find the appearance-based 2D correspondences, there are two approaches: matching and classification. Matching-based approaches [1, 3, 10, 11] use descriptors to store a signature for each model keypoint in a database. These descriptors are designed to be invariant to various geometric and photometric transformations, and can then be matched given a suitable distance metric to keypoints in an image in a nearest-neighbour fashion.

Classification-based approaches [9, 12, 15] treat matching as multi-class classification, in which the task is to classify each image keypoint as either background or a particular keypoint from the model. These classifiers are learned offline from training examples of the object observed under various geometric and photometric transformations (usually generated synthetically), and are therefore tuned to the specific object and how individual keypoints might appear under various illumination levels and new view-points. The training algorithm and the number of training examples determine the computational complexity of the learning stage.

Since classification-based approaches rely on the availability of a 2D/3D object model at training time, these approaches cannot easily be used for *on-the-fly* object detection and tracking. In other words, these algorithms are not suitable for detection and tracking of arbitrary unknown objects. This particular problem of the classification-based approaches limit their applicability in practice.

In [13], the authors propose an approach for learning a classification-based model at runtime, by using online random forests to reduce training time. However, this approach is still too computationally expensive to be useful on low-powered devices, and also does not continue to adapt the model after the initial training phase. The method presented in [6] is most related to our work, in which the authors learn keypoint classifiers online by using Haar features and an online boosting algorithm. This approach relies on the fact that the geometric verification step can be used in order to provide labels for updating the classifiers in an online manner, allowing for adaptive tracking by detection.

To the best of our knowledge, all previous methods involving learning treat the generation of correspondences and estimation of object transformation separately. In this paper, we propose a novel approach which combines these two steps into a coherent structured output learning framework. In this formulation, correspondence generation, learning, and transformation estimation are all working together in a unified optimization formulation with the goal of performing object detection robustly. Our approach proposes an alternative view on keypoint-based object detection where the transformation estimation algorithm operates as the maximization step of a structured output learning framework. Unlike the online boosting approach of [6], our formulation is also capable of incorporating any kind of keypoint descriptor into its learning process and is specifically targeted towards low-powered devices.

Structured output prediction was introduced to the computer vision community in [2] for the task of 2D sliding-window object detection. In [7], the authors use a similar approach with online learning to perform adaptive 2D tracking by detection. Our work differs from these approaches in that we are interested in object detection and tracking under a much larger class of transformations such as 3D pose or homography, and as a result we propose using RANSAC in order to perform structured output prediction.

There has recently been significant research interest focusing on object detection for low-powered portable platforms such as smartphones. In particular, highly efficient methods such as BRIEF [3] and BRISK [10] have been developed for descriptor matching. Both of these methods perform simple binary pixel-based tests on keypoints in order to build binary descriptors. By representing these descriptors as bitsets and measuring similarity using the hamming distance, matching can be performed extremely efficiently using bitwise operations which are well-supported by modern CPUs. We show how the internal representation of our algorithm can be approximated to take advantage of these binary descriptors, making our approach also suitable for low-powered devices.

3. Structured output formulation

In this section, we describe our formulation of keypoint-based object detection as a structured output problem.

3.1. RANSAC for structured prediction

Given an object model M and an input image I, the goal of object detection is to compute a transformation $T \in \mathcal{T}$ which maps M to I. A 3D pose or 2D homography are examples of such a transformation.

We can think of this process as one of structured output prediction, with the output space consisting of all valid transformations, along with a null transformation indicating the absence of the object. We therefore assume that there exists a function $T = f(M, I)$, and that this function can be expressed as

$$T = \underset{T' \in \mathcal{T}}{\operatorname{argmax}} F(M, I, T'), \tag{1}$$

where F is a compatibility function, scoring all possible transformations of the object given an image.

In practice, finding a solution for the prediction function Eq (1) under a specific model definition is generally infeasible because the output space is very large, and evaluating image observations under different transformations of the model will be expensive. The way that this issue is usually handled is by applying an iterative robust parameter estimation algorithm such as RANSAC [5] or PROSAC [4] to approximately solve Eq (1). These algorithms rely on a sparse representation for the model and image and use a set of correspondences between model and image points as their input.

Consider an object model M which is based on a sparse set of keypoints $\mathcal{M} = \{u_1, \ldots, u_J\}$, with each keypoint defined by a location (2D or 3D). Similarly, let the image I be represented as a sparse set of keypoints $\mathcal{I} = \{v_1, \ldots, v_K\}$. A set of correspondences $\mathcal{C} = \{(u_j, v_k, s_{jk}) | u_j \in \mathcal{M}, v_k \in \mathcal{I}, s_{jk} \in \mathbb{R}\}$ is found between model keypoints and image keypoints, where s_{jk} is a correspondence score derived from appearance information. Traditional RANSAC maximizes the number of inliers defined by

$$F(\mathcal{C}, T) = \sum_{(u_j, v_k) \in \mathcal{C}} \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau), \tag{2}$$

where $T(u_j)$ is the location of model keypoint u_j under the transformation T, τ is a spatial mis-alignment threshold and $\mathbb{I}(.)$ is an indicator function. This maximization is performed by randomly sampling transformations which are compatible with minimal subsets of correspondences in $\mathcal{C}$, with variants such as PROSAC biasing this sampling by using the correspondence scores s_{jk}.

Existing approaches have applied learning in an offline setting [9, 12, 15] as well as in an online setting [6, 13] to encourage reliable appearance-based correspondences to be found in $\mathcal{C}$. However, in these approaches the generation of correspondences and the scoring and maximization (Eq (2)) are decoupled from each other. These approaches therefore do not perform learning which takes into account the entire transformation prediction process.

To allow learning for the entire prediction process, we propose introducing a weight vector $\mathbf{w}_j$ for each model keypoint u_j. This weight vector is used to score correspondences according to $s_{jk} = \langle \mathbf{w}_j, \mathbf{d}_k \rangle$, where $\mathbf{d}_k$ is a descriptor extracted around image keypoint v_k, normalized such that $\|\mathbf{d}_k\|_2 = 1$. We then propose modifying the compatibility function Eq (2) to include correspondence scores, such that it can be written as a linear operator

$$\begin{aligned} F_{\mathbf{w}}(\mathcal{C}, T) &= \sum_{(u_j, v_k) \in \mathcal{C}} s_{jk}\, \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau) \\ &= \langle \mathbf{w}, \mathbf{\Phi}(\mathcal{C}, T) \rangle, \end{aligned} \tag{3}$$

where $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_J]^T$ is the concatenation of model weight vectors and $\mathbf{\Phi}(\mathcal{C}, T) = [\boldsymbol{\phi}_1(\mathcal{C}, T), \ldots, \boldsymbol{\phi}_J(\mathcal{C}, T)]^T$ is a joint feature mapping. Each $\boldsymbol{\phi}_j$ is defined as

$$\boldsymbol{\phi}_j(\mathcal{C}, T) = \begin{cases} \mathbf{d}_k & \exists (u_j, v_k) \in \mathcal{C} : \|v_k - T(u_j)\|_2 < \tau \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{4}$$

Our goal is to learn a model parameterized by $\mathbf{w}$ such that the behaviour of this function in the output space is close to the actual behaviour of RANSAC, but, because it includes information about appearance, in the process of learning we will discover which model points are the most discriminative and how best we can utilize them to predict transformations.

3.2. Structured output learning

Now, given a set of training examples $\{(I_i, T_i)\}_{i=1}^N$, $\mathbf{w}$ can be learned in a structured output maximum margin framework [17]. For each training example i, this formulation tries to maximize the margin between the score of the true output T_i and all alternative outputs. This can be expressed by the following optimization problem

$$\begin{aligned} \min_{\mathbf{w}, \boldsymbol{\xi}} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \forall i : \ \xi_i \geq 0 \\ & \forall i, \forall T \neq T_i : \ \delta F^i_{\mathbf{w}}(T) \geq \Delta(T_i, T) - \xi_i \end{aligned} \tag{5}$$

where $\delta F^i_{\mathbf{w}}(T) = F_{\mathbf{w}}(\mathcal{C}_i, T_i) - F_{\mathbf{w}}(\mathcal{C}_i, T)$, and λ is a parameter determining the trade-off between training set accuracy and regularization. $\Delta(T_i, T)$ is a loss function which measures the penalty for choosing T instead of the true transformation T_i. The loss function $\Delta(T_i, T)$ should measure the dissimilarity of two competing output hypotheses, and will be discussed in Section 3.3.

Because we are using RANSAC to perform the output prediction and this relies on an accurate set of correspondences, we modify this formulation to also encourage each

inlier correspondence to score higher than any other image correspondence. This can be realized as an additional set of ranking constraints and the formulation then becomes

$$\begin{aligned}\min_{\mathbf{w},\boldsymbol{\xi},\boldsymbol{\gamma}} \; & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N}\xi_i + \nu\sum_{i=1}^{N}\sum_{(u_j,v_k)\in\mathcal{C}_i^*}\gamma_{ij} \\ \text{s.t. } & \forall i:\ \xi_i \geq 0 \\ & \forall i, \forall T \neq T_i:\ \delta F_{\mathbf{w}}^i(T) \geq \Delta(T_i, T) - \xi_i \\ & \forall i, \forall j:\ \gamma_{ij} \geq 0 \\ & \forall i, \forall (u_j, v_k), \forall k' \neq k : \langle \mathbf{w}_j, \mathbf{d}_k - \mathbf{d}_{k'}\rangle \geq 1 - \gamma_{ij}\end{aligned} \tag{6}$$

where $\mathcal{C}_i^* \subset \mathcal{C}_i$ is the set of inlier correspondences under T_i, and ν is a weighting parameter.

The learning problem presented in Eq (6) allows us to train a discriminative model in a unified framework where learning the representation of model points and performing pose estimation is combined in a structured output maximum margin setting.

3.3. Loss function

Eq (6) requires a loss function Δ to be defined between two transformations. Since the compatibility function $F_{\mathbf{w}}(\mathcal{C}, T)$ sums over those correspondences in $\mathcal{C}$ which are inliers under T, we desire a loss function which takes into account the fact that transformations will have different numbers of inliers. We consider two such loss functions, which we compare experimentally in Section 4:

1. Hamming distance on inliers:

$$\Delta_H(T, T') = \sum_{(u_j,v_k)\in\mathcal{C}} \mathbb{I}\big(z(u_j, v_k, T) \neq z(u_j, v_k, T')\big) \tag{7}$$

where $z(u_j, v_k, T) = \mathbb{I}(\|v_k - T(u_j)\|_2 < \tau)$. This loss function aims to penalize transformations having different inlier sets.

2. Difference in number of inliers:

$$\Delta_I(T, T') = |F(\mathcal{C}, T) - F(\mathcal{C}, T')|. \tag{8}$$

This loss function aims to penalize transformations with different numbers of inliers, similar in spirit to the traditional RANSAC scoring function (Eq (2)).

3.4. Online learning

While Eq (6) can be solved offline as a batch problem, for our application we are interested in updating $\mathbf{w}$ online, such that we can adapt the model to a given environment. The model can be initialized by setting each $\mathbf{w}_j$ to be the descriptor one would use in a static model, and in subsequent frames can be updated by performing stochastic gradient descent. We rewrite the optimization problem of Eq (6) in unconstrained form as

$$\begin{aligned}\min_{\mathbf{w}}\Big\{ & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N}\big(\max_{T\neq T_i}\{\Delta(T_i, T) - \delta F_{\mathbf{w}}^i(T)\}\big)_+ + \\ & \nu\sum_{i=1}^{N}\sum_{(u_j,v_k)\in\mathcal{C}_i^*}\big(\max_{k'\neq k}\{1 - \langle \mathbf{w}_j, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\}\big)_+\Big\}\end{aligned} \tag{9}$$

where $(.)_+ = \max\{0, .\}$ is the hinge function.

Given a training example (I_t, T_t) at time t, a subgradient of Eq (9) is found with respect to $\mathbf{w}$, and a gradient descent step is then performed according to

$$\begin{aligned}& \mathbf{w}_j^{t+1} \leftarrow (1 - \eta_t\lambda)\mathbf{w}_j^t + \\ & \mathbb{I}(\max_{T\neq T_t}\{\Delta(T_t, T) - \delta F_{\mathbf{w}}^t(T)\} > 0)\eta_t\boldsymbol{\alpha}_j^t + \\ & \mathbb{I}(u_j \in \mathcal{C}_t^*)\ \mathbb{I}(\max_{k'\neq k}\{1 - \langle \mathbf{w}_j^t, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\} > 0)\eta_t\nu\boldsymbol{\beta}_j^t,\end{aligned} \tag{10}$$

where $\eta_t = 1/\lambda t$ is the step size. Let $\hat{T} = \operatorname{argmax}_{T\neq T_t}\{\Delta(T_t, T) - \delta F_{\mathbf{w}}^t(T)\}$ and $\hat{k} = \operatorname{argmax}_{k'\neq k}\{1 - \langle \mathbf{w}_j^t, \mathbf{d}_k - \mathbf{d}_{k'}\rangle\}$. Then $\boldsymbol{\alpha}_j^t$ and $\boldsymbol{\beta}_j^t$ are defined as

$$\boldsymbol{\alpha}_j^t = \boldsymbol{\phi}_j(\mathcal{C}_t, T_t) - \boldsymbol{\phi}_j(\mathcal{C}_t, \hat{T}), \tag{11}$$

and

$$\boldsymbol{\beta}_j^t = \mathbf{d}_k - \mathbf{d}_{\hat{k}}. \tag{12}$$

To estimate T_t for the current image, we use the prediction of Eq (1) using the old model representation $\mathbf{w}_{t-1}$ and then update the model according to Eq (10). Furthermore, when performing RANSAC in our prediction function we will also be exploring and scoring other transformations, which gives us a mechanism for identifying any margin violations which have occurred, the largest of which will contribute to the gradient descent step Eq (10). In this way, our online learning can re-use the intermediate results of estimating T_t, and thus adds only a small amount of overhead compared to detection alone.

3.5. Binary approximation of model

An important goal of our method is to be real-time and suitable for low-powered devices, and we would therefore like to take advantage of binary descriptors. Although these descriptors are very compact when represented as bitsets, to use a linear SVM requires converting them into high-dimensional real vectors. While this is acceptable when updating the classifier, it would be very computationally expensive at the matching stage, which requires exhaustive evaluation of every model classifier with every image keypoint. To avoid this, we propose approximating each $\mathbf{w}_j$ in

terms of a set of basis vectors

$$\mathbf{w}_j \approx \sum_{i=1}^{N_b} \beta_i \mathbf{b}_i \tag{13}$$

where $\mathbf{b}_i \in \{-1, 1\}^D$, and D is the dimensionality of the descriptor. This approximation must be updated each time $\mathbf{w}_j$ changes, so we choose to use a simple greedy method as described in Algorithm 1.

Algorithm 1 Binary approximation of $\mathbf{w}_j$

Require: $\mathbf{w}_j$, N_b
  $\mathbf{r} = \mathbf{w}_j$ *(initialize residual)*
  for $i = 1$ to N_b **do**
    $\mathbf{b}_i = \text{sign}(\mathbf{r})$
    $\beta_i = \langle \mathbf{b}_i, \mathbf{r} \rangle / \|\mathbf{b}_i\|^2$ *(project* $\mathbf{r}$ *onto* $\mathbf{b}_i$*)*
    $\mathbf{r} \leftarrow \mathbf{r} - \beta_i \mathbf{b}_i$ *(update residual)*
  end for
  return $\{\beta_i\}_{i=1}^{N_b}, \{\mathbf{b}_i\}_{i=1}^{N_b}$

Using this approximation, we can efficiently compute the dot-product $\langle \mathbf{w}_j, \mathbf{d} \rangle$ using only bitwise operations. To do so, we represent each $\mathbf{b}_i$ using a binary vector and its complement: $\mathbf{b}_i = \mathbf{b}_i^+ - \overline{\mathbf{b}_i^+}$, where $\mathbf{b}_i^+ \in \{0, 1\}^D$. We then rewrite

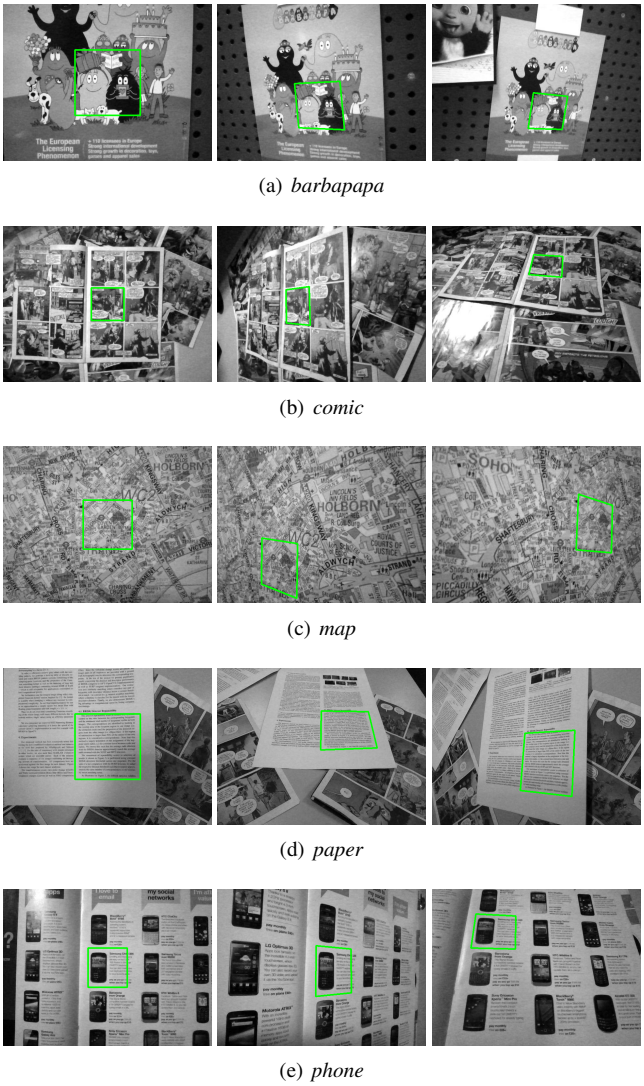
$$\langle \mathbf{w}_j, \mathbf{d} \rangle \approx \sum_{i=1}^{N_b} \beta_i (\langle \mathbf{b}_i^+, \mathbf{d} \rangle - \langle \overline{\mathbf{b}_i^+}, \mathbf{d} \rangle), \tag{14}$$

and note that each dot-product inside the summation can be computed very efficiently using a bitwise AND followed by a bit-count. This can be computed even more efficiently if we have precomputed the bit-count of $\mathbf{d}$, since $\langle \mathbf{b}_i^+, \mathbf{d} \rangle - \langle \overline{\mathbf{b}_i^+}, \mathbf{d} \rangle = 2\langle \mathbf{b}_i^+, \mathbf{d} \rangle - |\mathbf{d}|$. This means that by approximating $\mathbf{w}_j$ with N_b components, our correspondence score is roughly N_b times more expensive to evaluate than a binary Hamming distance. In practice, we find it sufficient to set $N_b = 2$, see Section 4 for experimental results.

4. Experiments

We performed a number of experiments in order to validate the approach described in this paper. Our method is applicable to general object models and transformations, but for the purposes of our experiments we consider the case of a planar object model detected in an image under a homography transformation.

We recorded a number of video sequences of a static scene observed from a moving camera, using a SLAM system to track the 3D camera pose in each frame (example frames can be seen in Figure 1). Each sequence begins with a fronto-parallel view of a planar patch, which is used in our experiments to define the object model. Using the known camera pose, we computed a ground-truth homography for the object in each video frame, which is then used for evaluating the quality of the homography estimates produced during object detection in our experiments.



(a) *barbapapa*

(b) *comic*

(c) *map*

(d) *paper*

(e) *phone*

Figure 1. Example frames from our test sequences, which also show the detection results for the BRIEF model learned using our structured output framework (Section 4.1). These sequences are challenging for keypoint-based matching approaches due to the presence of many similar features in the scene.

In order to evaluate the effectiveness of our approach and to observe the contribution of learning and the combined structured output framework, we also implemented a baseline of independently online trained SVM classifiers for each model keypoint. In this framework, we take away the coupling between model points that comes from our model and train each SVM classifier independently of one other. At run-time, we compute a matching score for the

Sequence	BRIEF				BRISK				SURF				Boost. [6]
	Static	Indep.	Δ_H	Δ_I	Static	Indep.	Δ_H	Δ_I	Static	Indep.	Δ_H	Δ_I	
barbapapa	0.19	**0.94**	**0.94**	**0.94**	0.92	0.93	**0.94**	**0.94**	0.89	0.45	**0.92**	**0.92**	0.88
comic	0.41	0.90	0.94	**0.98**	0.42	0.60	0.61	**0.76**	0.83	0.67	0.90	**0.93**	0.56
map	0.82	0.98	**0.99**	**0.99**	0.79	0.91	0.91	**0.93**	0.91	0.09	0.98	**0.99**	0.83
paper	0.06	0.68	0.77	**0.85**	0.04	0.40	0.51	**0.54**	**0.03**	0.01	**0.03**	**0.03**	0.04
phone	0.88	0.93	**0.97**	**0.97**	0.64	0.82	0.91	**0.92**	0.92	0.46	0.96	**0.97**	0.85

Table 1. Average detection rates in test sequences (the higher better). Each row represents a video sequence. Each set of columns shows a different combination of feature point detector and descriptor, while the last single column is the result of the boosting approach. Within a feature detector/descriptor combination, we compare the results of no learning (static), independently trained SVM classifiers, and our structured output learning framework with the two loss functions Δ_H and Δ_I defined in Section 3.3. The bold-face font represents the best working method for a video sequence in a chosen detector/descriptor setting.

j-th model keypoint using the learned SVM as $f_j(\mathbf{d}_k) = \langle \mathbf{w}_j, \mathbf{d}_k \rangle$ and use this score to find the highest scoring match to construct the correspondence set for pose estimation. To update each classifier, we take each inlier returned from the geometric verification set as a positive training example, and select the next highest scoring image keypoint as a negative example. We then perform stochastic gradient descent learning to update the classifiers.

Additionally, we implemented the boosting-based classification approach used in [6], by making use of the publicly available online boosting code of the authors[1]. We train these classifiers in the same manner as our independent SVM baseline.

The unoptimized C++ implementation of our approach as well as the annotated videos used during our experiments are publicly available to download[2].

4.1. Tracking by detection

To illustrate the benefit of online learning we consider the task of tracking-by-detection, in which the target object should be detected in consecutive frames of a video sequence. For this task we do not use any information about the location of the object in the previous frame when detecting the object, but we use each successful detection in order to perform an online learning step to update our object model for subsequent frames.

We apply our approach using three different combinations of interest point detector and descriptor: FAST detector with 256-bit BRIEF descriptor, BRISK detector with 512-bit BRISK descriptor and SURF detector with SURF64 descriptor. These have been chosen to illustrate that our method works with a variety of feature point detectors and descriptors, but as they each have different invariances and dimensionality, our results should not be interpreted as a comparison between different descriptor types. Therefore, we are interested in relative performance figures for a particular feature point detector and descriptor combination.

When learning with binary descriptors, we apply the feature transformation $\tilde{\mathbf{d}} = (\mathbf{d} - \mathbf{0.5})/0.5\sqrt{D}$, where D is the dimensionality of the descriptor, which centers and normalizes the descriptors, as this is known to improve the performance of stochastic gradient descent algorithms [8]. During matching this transformation can easily be handled implicitly in the binary approximation without any overhead. We fix the SVM learning rate $\lambda = 0.1$ for all experiments. We also set $\nu = 1$ for the structured model.

For each sequence, we initialize a model using the fronto-parallel planar patch in the first frame, by detecting the 100 strongest features to define the locations of model keypoints $\mathcal{M}$. Given these locations we initialize four models for comparison: a non-adaptive matching-based model using the descriptors for the model keypoints extracted from the first frame (to represent a traditional keypoint-based object detection approach); the baseline learned model using independent SVM classifiers; and our structured output approach with the two loss functions described in Section 3.3. We initialize the weight vector for a model keypoint by setting it to the descriptor from the first frame.

To assess detection performance, we define a scoring function between the ground-truth homography T^* and the predicted homography T as:

$$S(T^*, T) = \frac{1}{4} \sum_{i=1}^{4} \|\mathbf{c}_i - (T^* T^{-1})(\mathbf{c}_i)\|_2, \qquad (15)$$

where $\{\mathbf{c}_i\}_{i=1}^4 = \{(-1,-1)^\mathsf{T}, (1,-1)^\mathsf{T}, (-1,1)^\mathsf{T}, (1,1)^\mathsf{T}\}$ define the corners of a square. This score will be 0 if the two homographies are identical. Frames for which $S(T^*, T) < 10$ are considered correct detections, and we report the average over the entire sequence.

The results of our experiments without the binary approximation described in Section 3.5 can be seen in Table 1[3]. As can be seen from this table, the structured output learning framework outperforms the static model (with

[1] http://www.vision.ee.ethz.ch/boostingTrackers/onlineBoosting.htm
[2] http://www.samhare.net/research

[3] The actual videos and the results of tracking-by-detection can be found in our supplementary material or at http://www.samhare.net/research.

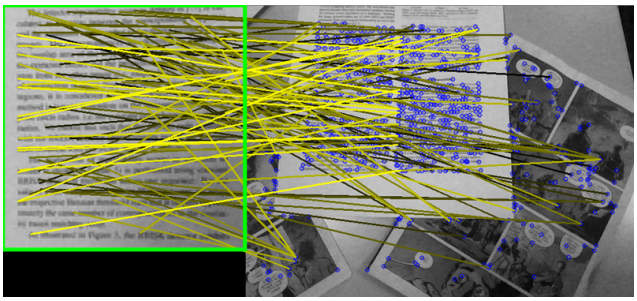
no learning), as well as the model trained with independent SVM classifiers. Comparing the results of independent SVM classifiers and the static model highlights the fact that adapting an object model to a particular environment online helps a lot in practice. However, the highest detection rate is attained when we used the structured output framework where the learning of the object model and geometric estimation is linked inside a unified optimization formulation. In particular, using the loss function Δ_I based on the difference in number of inliers results in the best performance. It should be noted that for SURF descriptors the independent SVMs had difficulty learning the correct object model. We suspect that this is caused because of the continious nature of the SURF descriptor and the fact that the number of generated keypoints is lower with the SURF keypoint detector. However, given the same settings, the structured learning approch is able to benefit fully from the adaptation process and improve upon the static model.

For the boosting-based online descriptor learning approach, it is only fair to compare against the models where we use the BRIEF descriptor (as both of these methods are using the same FAST corner detector). Again one can see that comparing the boosting method with the static method, learning still provides an improvement. However, the boosting based approach is not able to outperform the independent SVM learning framework, and is therefore also performing worse than the structured output framework.

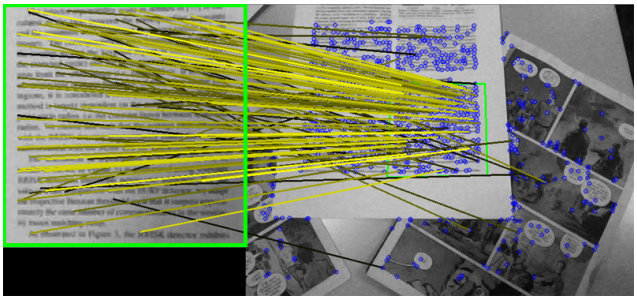
The most difficult video in our set of experiments is the *paper* sequence. This video sequence features highly repetitive local appearance structures and a simple static model fails in all cases. The learning based approaches (except the boosting method) are able to deliver a reasonable detection rate using binary descriptors. An example frame of this sequence is shown in Figure 2 where we display the generated correspondences before geometric verification. As can be seen in the top image, because of the confusing appearance of the local image features, the static BRIEF model fails to match model keypoints consistently to the image. However, the structured learning framework which uses the same set of descriptors extracted from the input image for matching has learned a more discriminative object model and is able to provide more meaningful correspondences which are mainly focused on where the object is in the input frame. Another observation is that although the structured learning model creates some mis-matches (lines matching an object point to some background locations), they all contain very small matching scores (indicated by their dark color).

4.2. Binary approximation

To verify that the binary approximation proposed in Section 3.5 is reasonable when using binary descriptors such as BRIEF and BRISK, we repeat our experiments for the BRIEF descriptor model learned in our structured output framework and approximate the model keypoint weight vectors $\mathbf{w}_j$ with varying numbers of binary bases N_b. As can be seen in Figure 3, in general the binary approximation produces detection performance comparable to the original classifier with $N_b \geq 2$ bases, and for the less challenging sequences even a single basis suffices. In terms of detection time, which includes the stages of generating correspondences between model and image, performing geometric verification, and updating the learner, we see that the binary approximation provides significant performance gains (approximately 4 times faster detection with our unoptimized implementation). This means that our approach is suitable for use even on low-powered devices.



(a) Static BRIEF model

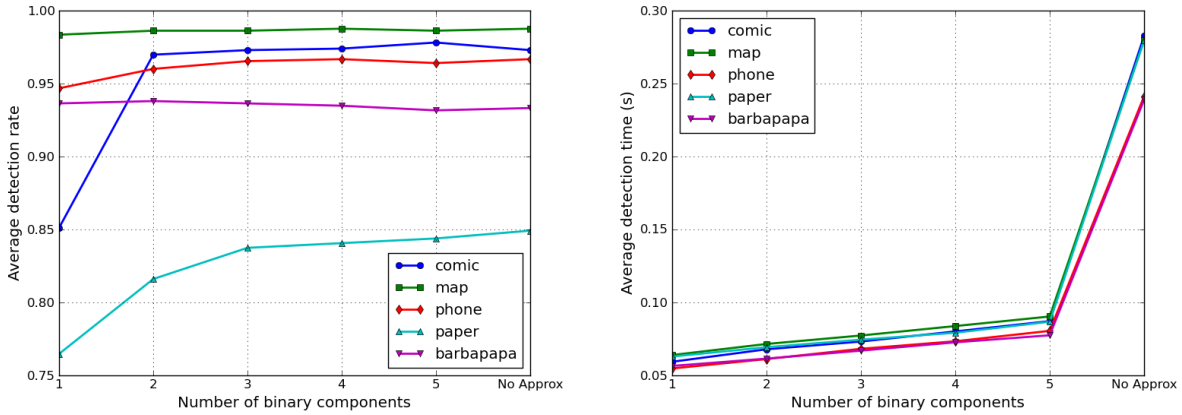


(b) Learned BRIEF model using our structured output formulation

Figure 2. Example frames from the *paper* sequence showing the top correspondence for each model point. The model is displayed in a green box on the left of this image. The brightness of each line indicates the correspondence score, before any geometric verification has taken place (the brighter the higher the score). The learned model has adapted to discriminate against the many confusing keypoints in the image, resulting in a successful detection, while no detection is found with the static model.

5. Conclusions

In this paper, we presented a novel approach to learning for real-time keypoint-based object detection and tracking. Our formulation generalizes previous methods by combining the feature matching, learning, and object pose estimation into a coherent structured output learning framework. We showed how such a model can be trained on-



(a) Detection rate

(b) Detection time

Figure 3. Behaviour of the learned BRIEF model using our structured output formulation when employing a binary approximation of each $\mathbf{w}_j$ as described in Section 3.5. For $N_b \geq 2$ the detection performance is almost equivalent to the original model, whilst being approximately four times faster with our unoptimized implementation.

line, and presented an approximation to create an efficient way of using binary descriptors at runtime. During our experiments we showed that learning in the unified structured output learning formulation plays an important role in improving the detection rate compared to state-of-the-art static and learning-based feature matching techniques.

While we did not perform feature selection explicitly, our formulation implicitly is able to down-weight the less discriminative features, and therefore, provides a good starting platform for further research into automatic online feature selection.

Acknowledgements This work is supported by EPSRC CASE and KTP research grants and the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. P. H. S. Torr is in receipt of Royal Society Wolfson Research Merit Award.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *CVIU*, 110(3):346–359, June 2008.

[2] M. B. Blaschko and C. H. Lampert. Learning to Localize Objects with Structured Output Regression. In D. Forsyth, P. Torr, and A. Zisserman, editors, *ECCV*, volume 5302 of *Lecture Notes in Computer Science*, pages 2–15, Berlin, Heidelberg, Oct. 2008. Springer Berlin Heidelberg.

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *ECCV*, 2010.

[4] O. Chum and J. Matas. Matching with PROSAC Progressive Sample Consensus. In *CVPR*, 2005.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[6] M. Grabner, H. Grabner, and H. Bischof. Learning Features for Tracking. In *CVPR*, 2007.

[7] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured Output Tracking with Kernels. In *ICCV*, 2011.

[8] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient BackProp. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, page 546. Springer Berlin / Heidelberg, 1998.

[9] V. Lepetit and P. Fua. Keypoint Recognition Using Randomized Trees. *PAMI*, 28(9):1465–79, Sept. 2006.

[10] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *ICCV*, 2011.

[11] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, Nov. 2004.

[12] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast Keypoint Recognition Using Random Ferns. *PAMI*, 32(3):448–61, Mar. 2010.

[13] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature Harvesting for Tracking-by-Detection. In *ECCV*, 2006.

[14] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[15] S. Taylor and T. Drummond. Multiple Target Localisation at over 100 FPS. In *BMVC*, 2009.

[16] P. H. S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, Apr. 2000.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484, Dec. 2005.